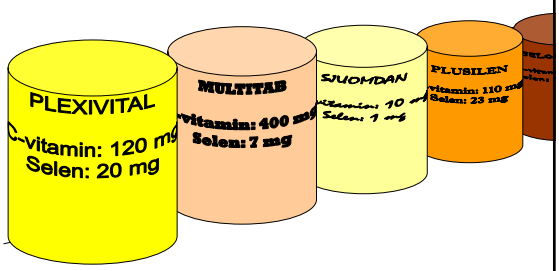
Föreläsning 9

Sökning, sortering, grafiska gränssnitt (GUI)

Schemaändringar

- **Tisdag 10 april:**
  - Ingen föreläsning 8-10
  - Labb 10-12
  - Labb 13-15
- **Tisdag 17 april**
  - Labb 10-12
  - Föreläsning 13-15 i sal E1

Vitaminexempel



Vitaminklassen

```
class Vitamin(object):
    """Representerar en sorts vitaminer"""

    def __init__(self, namn, c, selen):
        """Konstruktorn"""
        self.c = c
        self.selen = selen
        self.namn = namn

    def __str__(self):
        """För utskrift"""
        return self.namn +
               "\n C-vitamin: " + str(self.c) +
               "\n Selen: " + str(self.selen)
```

Vitaminer i lista

```
# Skapa tre Vitamin-Objekt
v1 = Vitamin("Plexivital",120,20)
v2 = Vitamin("Multitab",400,7)
v3 = Vitamin("Sjuomdan",10,1)

# Lägg objekten i en lista
lista = [v1,v2,v3]
```

Algoritm för sökning

1. Skapa en tom funna-lista för alla träffar.
2. Gå igenom varje element i vitamin-listan
   - Om elementet är det sökta:
     Lägg till vitaminen till funna-listan
3. Efteråt har man en funna-lista med alla träffar.

Funktion för sökning

```
def sok(x,lista):
    """Söker efter objekt som heter x """
    funna = []
    for v in lista:
        if v.namn == x:
            funna.append(v)
    return funna
```

Testa sökfunktionen

```
funna = sok("Sjuomdan",lista)

for v in funna:
    print v
```

Sortering

- Python-listor har en inbyggd sort-metod:
  - `lista.sort()`
- Men om man har en lista av objekt - vilket attribut sorterar `sort()` på då?
- Och tänk om man vill sortera på olika attribut, t ex först på c-vitamin-innehåll och sedan på selen-innehåll?

Lägg till get-metoder i klassen

```
def get_c(self):
    """Ger C-vitamin-innehåll"""
    return self.c

def get_selen(self):
    """Ger selen-innehåll"""
    return self.selen

def get_namn(self):
    """Ger namnet"""
    return self.namn
```

Sort med key

- Nu kan sort anropas med parametern key satt till get-metoden för det vi vill sortera på!

```
lista.sort(key=Vitamin.get_c)
```

Grafiskt användargränssnitt (GUI)

Använd modulen **Tkinter**, som har klasser för *komponenter*.

Se "Referenser" på kursens webbsida:
GUI-länkar.

Button	Menu
Canvas	Menubutton
Checkbutton	Message
Entry	Radiobutton
Frame	Scale
Label	Scrollbar
Listbox	Text

Hur gör man?

```
from Tkinter import *
roten = Tk()
knapp = Button(roten, text="Tryck")
knapp.grid()
roten.mainloop()
```

Hämtar alla klasser i modulen Tkinter.

Tk-konstruktorn - skapar rotfönstret.

Button-konstruktorn - skapar en knapp.

Knappen placeras.

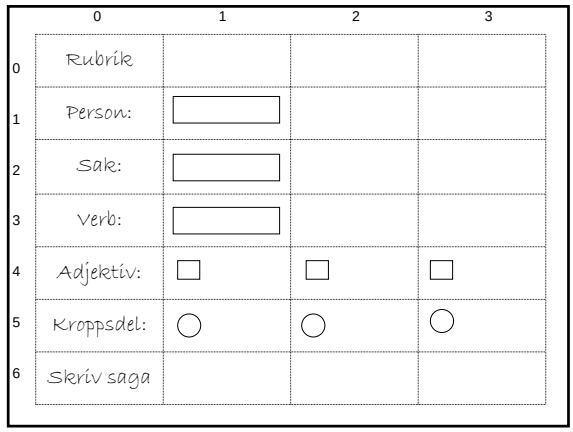
Startar en slinga som väntar på inmatning från användaren.

Komponenter

- Varje komponent har en konstruktor med många defaultparametrar.
- Anropa bara med det som behövs:
  `knapp = Button(roten,text="Tryck")`
- Efter att komponenten skapats kan attributen också ändras:
  `knapp["text"] = "Tryck inte"`

Layout

- Komponenter har metoder som styr hur de ska placeras i fönstret.
- Enklast är att använda pack:
  `knapp.pack()`
- Bättre kontroll fås med grid:
  `knapp.grid(row=4, column=3)`
- Rita först en skiss över hur det ska se ut!



Variabler

- Ett attribut som alla komponenter har är `variable`.
- Om man i förväg skapat ett variabelobjekt:
  `str = StringVar()`
- så kan man koppla ihop variabel och komponent med
  `knapp["variable"] = str`
- Metoden `get` hämtar data från en variabel.

Anropa metod

- Ett annat attribut som alla komponenter har är `command`.
- Där anger man vilken metod/funktion som ska anropas när komponenten används.
- Om vi skriver en funktion addera() som ska anropas när nån trycker på knapp så kopplar vi ihop metod med knapp så här:
  `knapp["command"] = addera`